Rok akademicki 2012/2013

Politechnika Warszawska

Wydział Elektroniki i Technik Informacyjnych

Instytut Informatyki



praca dyplomowa magisterska

inż. Bartłomiej Jańczak

Tytuł pracy dyplomowej

Opiekun pracy

prof. nzw. dr hab. Marzena Kryszkiewicz

Ocena:

Podpis Przewodniczącego

Komisji Egzaminu Dyplomowego

Kierunek: Informatyka

Specjalność: Inżynieria Systemów Informatycznych

Data urodzenia: 1988.03.06

Data rozpoczęcia studiów: 2007.10.01

Życiorys

Podpis studenta

EGZAMIN DYPLOMOWY

Złożył egzamin dyplomowy w dniu 20\_\_ r

z wynikiem

Ogólny wynik studiów:

Dodatkowe wnioski i uwagi Komisji:

STRESZCZENIE

Streszczenie pracy w języku polskim.

Słowa kluczowe:

THESIS TITLE IN ENGLISH

Summary in English.

Keywords:

Spis treści

[1. Wprowadzenie 1](#_Toc351220958)

[1.1. Przegląd literatury 2](#_Toc351220959)

[1.2. Motywacja i cel pracy 2](#_Toc351220960)

[1.3. Układ pracy 3](#_Toc351220961)

[2. Miary odległości i podobieństwa 3](#_Toc351220962)

[2.1. Metryki odległości 3](#_Toc351220963)

[2.2. Miara odległości kosinusowej (1) 4](#_Toc351220964)

[2.3. Wyznaczanie kosinusowego sąsiedztwa za pomocą sąsiedztwa opartego na odległości euklidesowej? 5](#_Toc351220965)

[3. Użyte algorytmy 7](#_Toc351220966)

[3.1. Grupowanie gęstościowe na przykładzie DBSCAN 8](#_Toc351220967)

[3.2. Porównanie algorytmów VP-Tree i TI-k-Neighborhood 13](#_Toc351220968)

# 

# 1. Wprowadzenie

Współczesne systemy komputerowe agregują i generują ogromną ilość danych zawierających cenną dla biznesu trudno odkrywalną wiedzę. Jej znajdowaniem zajmuje się dziedzina informatyki zwana odkrywaniem wiedzy. Mimo, że jest ona stosunkowo młoda, to stworzyła wiele technik eksploracji danych, które dzięki swojej skuteczności oraz wydajności znalazły szerokie praktyczne zastosowanie w rozwiązywaniu problemów związanych z szeroko pojętą analizą danych.

Zasadniczą przyczyną rychłego rozwoju odkrywania wiedzy jest spopularyzowanie wydajnych metod pozyskiwania i gromadzenia informacji. Zjawisko to nie byłoby możliwe bez postępu technologicznego w dziedzinie urządzeń agregujących dane i systemów bazodanowych oraz dzięki upowszechnieniu urządzeń umożliwiających automatyczną rejestrację sposobu ich wykorzystania. Z punktu widzenia konsumenta można tu wyszczególnić kody kreskowe, karty płatnicze oraz szeroko pojęte urządzenia mobilne. Kolejnym wartym uwagi źródłem danych jest sieć Internet, w której możliwa jest rejestracja wielu czynności korzystających z niej użytkowników, którzy ponadto dobrowolnie umieszczają w niej wiele informacji o sobie, przykładowo na portalach społecznościowych.

Wyżej wymienione zjawiska mają wpływ na osiąganie ogromnych rozmiarów przez współczesne zbiory danych. Tempo ich wzrostu jest szybsze niż przewidywano jeszcze kilka lat temu. Ich gromadzenie i przechowywanie na nośnikach pamięci masowej nie stanowi problemów dla współczesnych systemów, natomiast działanie na takiej ilości danych, pomimo stale wzrastającej mocy obliczeniowej komputerów, wciąż jest wyzwaniem dla dzisiejszej informatyki. Zbiory danych same w sobie nie stanowią wielkiej wartości, jednakże rozsądnie wykorzystane mogą stać się cennym źródłem szczególnej wiedzy. Nierzadko użyteczna wiedza ukryta jest między pewnymi składowymi danych, więc do jej odkrycia konieczne są właściwe algorytmy. Zagadnieniom tym poświęcona jest dziedzina informatyki zwana eksploracją danych, której ideą jest wykorzystanie komputera do znajdowania ukrytych dla człowieka wartościowych prawidłowości w danych zgromadzonych w dużych repozytoriach.

Odkrywanie wiedzy jest procesem złożonym, najczęściej składają się na niego następujące etapy:

* analiza danych – poznanie charakteru danych i określenie celu eksploracji,
* selekcja danych – czyszczenie, weryfikacja poprawności i wybór danych, które zostaną poddane dalszej analizie,
* transformacja danych – przekształcenie danych do odpowiedniej postaci, określenie strategii wobec danych niepełnych,
* eksploracja danych – ekstrakcja wiedzy z danych,
* interpretacja wyników – logiczna i graficzna wizualizacja wyników, wybór najbardziej interesującej wiedzy, wnioskowanie.

Kluczową fazą procesu odkrywania wiedzy jest eksploracja danych. Do zasadniczych metod eksploracji danych należą:

* grupowanie,
* klasyfikacja,
* odkrywanie asocjacji,
* regresja.

Każda z metod ujawnia różnego rodzaju korelacje pomiędzy danymi, z czego wynika ich odmienne zastosowanie.

W tej pracy skoncentrowałem się na zagadnieniu grupowania danych, które określane jest jako wyznaczanie zbiorów obiektów podobnych przy zachowaniu właściwości maksymalizacji podobieństwa obiektów należących do tych samych grup i minimalizacji podobieństwa obiektów należących do innych grup.

## 1.1. Przegląd literatury

Grupowanie danych jest popularną metodą o wielu zastosowaniach, dlatego nie trudno o jej opis w literaturze. W przypadku algorytmów, na których skupiłem się w niniejszej pracy wyjątkowo przydatne okazały się artykuły naukowe.

Prawdopodobnie najpopularniejszym algorytmem gęstościowego grupowania danych jest DBSCAN ?? stanowiący często punkt odniesienia dla porównań z innymi algorytmami gęstościowych grupowań. [TODO]

Nową koncepcją zwiększenia wydajności wyżej wymienionych algorytmów jest wykorzystanie nierówności trójkąta do redukcji liczby kosztownych operacji wyznaczania podobieństwa obiektów. Na przykładzie algorytmu k-środków przedstawiane już były próby wykorzystania nierówności trójkąta w algorytmach grupowania danych. Natomiast po raz pierwszy została ona użyta w celu porządkowania dostępu do danych w algorytmach gęstościowego grupowania TI-DBSCAN ??, TI-NBC i PreDeCon. Dokonano również badania wpływu liczby punktów referencyjnych i strategii ich wyboru na efektywność tych algorytmów ??.

## 1.2. Motywacja i cel pracy

Grupowanie danych to proces powszechnie stosowany w porządkowaniu produktów, segmentacji klientów, organizacji obiektów czy rozpoznawaniu i nanalizie obrazów. Procesy te wymieniane są pośród kluczowych elementów, na których bazuje szeroko rozumiana sztuczna inteligencja. We współczesnym świecie algorytmy grupowania danych znajdują coraz szersze zastosowanie. Ich popularność rozpala zainteresowanie naukowców, którzy opracowują coraz sprawniejsze algorytmy lub modyfikują istniejące, które dotychczas wydawały się optymalne. Nierzadko zdarza się, że usprawnienia po wielokroć zwiększają wydajność dotychczasowych rozwiązań, co z kolei umożliwia przetwarzanie zbiorów danych z większą liczbą obiektów bądź atrybutów. Niekiedy może to oznaczać sposobność użycia tych algorytmów w nieosiągalnych dotychczas obszarach.

Jednym z najnowszych pomysłów na zwiększenie wydajności algorytmów grupowania danych jest zastosowanie nierówności trójkąta. [TODO]

Celem pracy jest … [TODO]

## 1.3. Układ pracy

Po wprowadzeniu w zagadnienia grupowania danych, gruntownie przestawiłem algorytm DBSCAN. Opis cech charakterystycznych algorytmu oraz specyficznej taksonomii zostały uzupełnione o pseudokody, do których odwołuję się w kolejnych rozdziałach, co pozwala spójnie i precyzyjnie przedstawić zmiany, które wprowadzone są w algorytmie w związku z wykorzystaniem nierówności trójkąta. Teoretycznie podstawy wprowadzanych modyfikacji przedstawiłem na początku rozdziału trzeciego.

# 2. Miary odległości i podobieństwa

Podobieństwo jest pojęciem fundamentalnym w niemal każdej dziedzinie naukowej. Przykładowo, w matematyce, geometryczne metody oceny podobieństwa wykorzystywane są do określania przystawania jak również w dziedzinach pokrewnych takich jak trygonometria. W biologii molekularnej ważnym problemem jest mierzenie podobieństwa par białek. Zbiory rozmyte wykształciły własne miary podobieństwa znajdujące zastosowanie na polach zarządzania, medycyny czy meteorologii. Przegląd wszystkich zastosowań podobieństwa jest samym w sobie wdzięcznym tematem na pracę dyplomową. W niniejszym rozdziale skupię się na wybranych miarach podobieństwa wektorów, tj. metrykach odległości oraz metryce podobieństwa kosinusowego.

## 2.1. Metryki odległości

Metryką odległości (lub krócej odległością) w zbiorze wektorów jest miara podobieństwa , która spełnia następujące warunki dla dowolnych wektorów , oraz w :

1. ;
2. ;
3. ;

Warto zauważyć, że istnieje wiele miar odległości. W zależności od zastosowania, jedne miary mogą być stosowniejsze niż inne w danym przypadku. Najpopularniejszą miarą odległości jest *odległość Euklidesowa*. *Odległość Euklidesowa* między punktami i oznaczana jest jako i definiowana w następujący sposób:

Jeśli stosowana jest odległość Euklidesowa to otoczenie punktu przyjmuje sferyczny kształt.

Innym przykładem popularnej miary odległości jest *odległość Manhattan*. *Odległość Manhattan* między punktami i oznaczana jest jako i definiowana w następujący sposób:

Jeśli stosowana jest odległość Manhattan to otoczenie punktu przyjmuje prostokątny kształt.

Zarówno odległość Manhattan jak i odległość Euklidesowa są szczególnymi przypadkami *odległości Minkowskiego*. *Odległość Minkowskiego* między punktami i oznaczana jest jako i definiowana w następujący sposób:

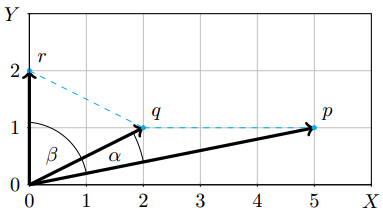
Dla uproszczenia, bez straty ogólności, w dalszych rozważaniach jako metryką odległości będę posługiwał się odległością euklidesową.

## 2.2. Miara odległości kosinusowej (1)

W wielu aplikacjach odkrywających wiedzę w danych tekstowych stosowana jest *miara podobieństwa kosinusowego* będąca funkcją konta między dwoma wektorami. *Miara podobieństwa kosinusowego* między wektorami i oznaczana jest jako i zdefiniowana w następujący sposób:

**Przykład 1**. Na rysunku Rys. 1 przedstawiono trzy wektory , i . Należy zwrócić uwagę, że odległość między i jest większa niż odległość euklidesowa między i . Z drugiej strony, w sensie podobieństwa kosinusowego, jest bardziej podobne do niż , ponieważ kosinus konta między i () jest większy niż kosinus konta między i ().

W tabeli Tab. 1 zamieszczono podobieństwa kosinusowe między wektorami , i . Można zauważyć, że dla podobieństwa kosinusowego warunki oraz i są niespełnione.



**Rys. 1.** Odległość euklidesowa i podobieństwo kosinusowe

**Tab. 1.** Podobieństwo kosinusowe wektorów z rys. 1.

|  |  |
| --- | --- |
| **(u,v)** | **cosSim(u,v)** |
| (p,q) | 0,965 |
| (p,r) | 0,196 |
| (q,r) | 0,447 |

Z powyższego przykładu płyną następujące wnioski:

1. Nierówność trójkąta nie jest spełniona dla dla dowolnego zbioru wektorów.
2. Nierówność trójkąta nie jest spełniona dla dla dowolnego zbioru wektorów.
3. Nierówność trójkąta nie jest spełniona dla dla dowolnego zbioru wektorów.

Ponieważ podobieństwo kosinusowe między niezerowymi wektorami i opiera się wyłącznie na kącie zawartym między nimi i nie zależy od ich długości, stąd obliczanie może być wyznaczone w oparciu o znormalizowane wektory i , tj. i . Z powyższego spostrzeżenia wynikają następujące własności:

1. ;
2. ;
3. .

## 2.3. Wyznaczanie kosinusowego sąsiedztwa za pomocą sąsiedztwa opartego na odległości euklidesowej?

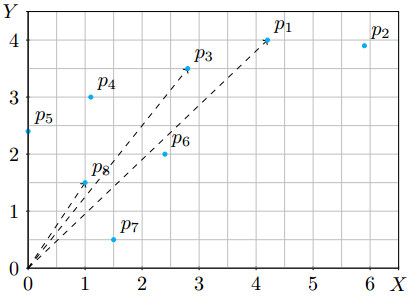
Artykuł (1) dowodzi, że kosinusowe sąsiedztwo[[1]](#footnote-2) w zbiorze wektorów może zostać wyznaczone za pomocą odpowiedniego sąsiedztwa opartego na odległości euklidesowej w zbiorze wektorów składającym się z α-znormalizowanych[[2]](#footnote-3) wektorów z . Stąd, autorka artykułu proponuje następujące podejście do wyznaczania sąsiedztwa opartego na podobieństwie kosinusowym.

W pierwszej kolejności początkowy zbiór wektorów transformowany jest do - zbioru α-znormalizowanych wektorów z . Następnie kosinusowe -sąsiedztwo (lub kosinusowe k-sąsiedztwo) w zbiorze ustanawiane jest jako oparte na odległości euklidesowej -sąsiedztwo (lub alternatywnie k-sąsiedztwo) w zbiorze , gdzie . Warto zwrócić uwagę, że w przeciwieństwie do kosinusowego -sąsiedztwa, -sąsiedztwo spełnia własność nierówności trójkąta.

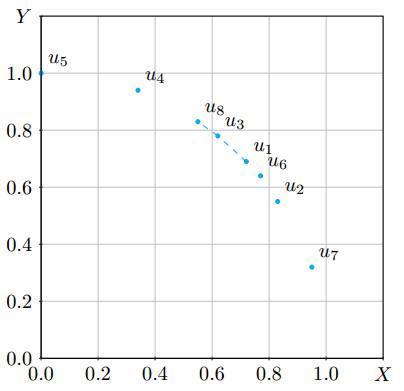
**Przykład 2.** W Tab. 2 zdefiniowano a na Rys. 2 przedstawiono przykładowy zbiór . Na Rys. 3 zamieszczono zbiór α-znormalizowanych wektorów z , gdzie . Warto zauważyć, że znormalizowany wektory zbioru mają długość równa , a punkty opisywane wektorami zbioru D’ układają się na okręgu o środku w punkcie i promieniu .

**Tab. 2.** Przykładowy zbiór wektorów

|  |  |  |
| --- | --- | --- |
| **Wektor** |  |  |
|  | 4.20 | 4.00 |
|  | 5.90 | 3.90 |
|  | 2.80 | 3.50 |
|  | 1.10 | 3.00 |
|  | 0.00 | 2.40 |
|  | 2.40 | 2.00 |
|  | 1.50 | 0.50 |
|  | 1.00 | 1.50 |



**Rys. 2.** Przykładowy zbiór wektorów



**Rys. 3.** Zbiór wektorów α-znormalizowanych

Załóżmy, że chcemy znaleźć wektory najbardziej podobne do wektora względem kosinusowego podobieństwa. Z Rys. 2 widać, że są nimi wektory i . Analogicznie przedstawia się relacja między α-znormalizowanymi wektorami , i , tj. , i . Najbardziej podobnymi wektorami do względem kosinusowego podobieństwa są wektory i . Co więcej wektory i są również najbardziej podobne do względem podobieństwa opartego na odległości euklidesowej, co zaznaczono na Rys. 3 błękitną przerywaną linią.

# 3. Użyte algorytmy

Istnieje wiele rozwiązań problemu grupowania danych czyli wyznaczania zbiorów obiektów podobnych przy zachowaniu właściwości maksymalizacji podobieństwa obiektów należących do tych samych grup i minimalizacji podobieństwa obiektów z różnych grup. Popularnym przykładem miary podobieństwa jest odległość Euklidesowa klasyfikująca obiekty leżace blisko siebie jako podobne, jednak większość algorytmów jest niezależna od przyjętej miary podobieństwa. Liczność zastosowań grupowania częstokroć o odmiennych wymaganiach co do rezultatu oraz specyficznych danych wejściowych (np. o różnej liczności, rozkładzie bądź liczbie atrybutów) prowadzi do dużej liczby wyspecjalizowanych algorytmów. W każdym z nich można doszukać się wad oraz zalet, jednakże nie znaleziono dotychczas uniwersalnego algorytmu. Często trudno porównywać algorytmy grupowania danych ponieważ ze względu na charakterystyczne podejście do rozwiązywanego problemu różnią się one nie tylko sposobem grupowania ale także definicją grupy.

Najpopularniejsza klasyfikacja algorytmów grupowania dzieli je na algorytmy oparte na podziale i algorytmy hierarchiczne. W przypadku pierwszej klasy kluczowym elementem jest znalezienie najlepszego podziału zbioru na z góry zadaną liczbę możliwie najbardziej jednorodnych grup. Początkowy podział odpowiednio ze zdeterminowaną strategią optymalizowany jest w kolejnych iteracjach zgodnie z przyjętą funkcją celu. Przykładami metod podziału są algorytmy k-średnich i k-medoidów. Wynikiem drugiej klasy algorytmów grupowania jest dendrogram[[3]](#footnote-4) - drzewo, które iteracyjnie dzieli zbiór danych na coraz to mniejsze podzbiory dopóki każdy podzbiór składa się z jednego obiektu. W takiej hierarchii każdy węzeł drzewa reprezentuje klaster zbioru danych. Relacja między węzłami a ich przodkami w dendrogramie odpowiada relacji między podgrupami a grupami. Dendrogramy mogą być tworzone od liści w górę do korzenia (*podejście aglomeracyjne*) lub od korzenia w dół do liści (*podejście podziału*) poprzez scalanie lub podział klastrów z każdym krokiem algorytmu. Obie wymienione klasy algorytmów grupowania posiadają pewne wady. W przeciwieństwie do algorytmów opartych na podziale algorytmy hierarchiczne nie oczekują arbitralnie zadanej liczby klastrów, jednakże wymagają zdefiniowania *warunku zakończenia* wskazującego kiedy proces podziału lub scalania powinien się zakończyć.

Wyniki wyżej wymienionych metod rzadko odpowiadają oczekiwaniom. Taki stan rzeczy można tłumaczyć nienaturalnym dla człowieka mechanizmem grupowania. Gdyby zadać człowiekowi zadanie pogrupowania punktów dwuwymiarowej przestrzeni okazałoby się, że nie dzieliłby on zbioru hierarchicznie na kolejne podzbiory czy też nie próbowałby podzielić go na z góry określoną liczbę podzbiorów. Ludzie z łatwością rozpoznają klastry o dowolnych kształtach oraz szum. Głównym powodem, dla którego rozpoznajemy klastry jest fakt, iż wewnątrz każdego z klastrów można wyszczególnić pewną gęstość punktów znacznie wyższą niż poza klastrem. Zatem do grupy należą punkty leżące w obszarze o gęstości wyraźnie większej niż w obszarze otaczającym ją. Tak zdefiniowanemu pojęciu metody grupowania najbliżej jest algorytmom gęstościowym, których przykładem jest DBSCAN opisany w kolejnym rozdziale.

## 3.1. Grupowanie gęstościowe na przykładzie DBSCAN

DBSCAN czyli Density Based Spatial Clustering of Applications with Noise jest jednym z najpopularniejszych algorytmów gęstościowego grupowania danych. Zaproponowany w 1996 roku wciąż jest sztandarowym algorytmem gęstościowym będącym punktem odniesienia w wielu pracach naukowych dotyczących tematyki grupowania oraz prezentujących nowe rozwiązania lub algorytmy.

Oprócz grup, czyli zbioru punktów o dużej gęstości punktów DBSCAN rozpoznaje również szum, do którego należą punkty leżące w obszarze o małej gęstości. Algorytm wymaga podania jedynie dwóch parametrów wejściowych, które opisują najmniejszy klaster będący obiektem zainteresowania. Jest to promień wokół danego punktuy, wewnątrz którego znajduje się minimalna liczba punktów. Para parametrów i stanowi intuicyjną definicję najmniejszej gęstości tym samym definiując minimalną liczność wykrywanych grup.

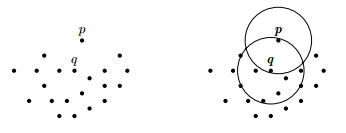
Podstawowym pojęciem używanym w kontekście algorytmu DBSCAN jest *otoczenie epsilonowe* oznaczane przez i definiowane jako zbiór takich punktów zbioru D, które są różne od i nie bardziej odległe od niż , czyli:

W DBSCAN wyróżnia się dwa rodzaje punktów wchodzące w skład klastra: punkty wewnątrz grupy zwane *punktami rdzeniowymi* oraz punkty leżące na obrzeżach klastra – *punkty brzegowe*.

*Punktem rdzeniowym* nazywamy taki punkt p, którego otoczenie epsilonowe zawiera wymaganą liczbę punktów, czyli:

Mówimy, że punkt jest *bezpośrednio gęstościowo osiągalny* z punktu względem i , jeżeli należy do otoczenia epsilonowego p, oraz q jest punktem rdzeniowym:

Relacja *bezpośredniej gęstościowej osiągalności* jest symetryczna tylko dla punktów rdzeniowych. Na Rys. 4 przedstawiono klaster, w którym zaznaczono pewien punkt brzegowy , punkt rdzeniowy , okręgami zaznaczono otoczenie epsilonowe równe a wynosi 5. Rysunek prezentuje asymetryczny przypadek, w którym jest bezpośrednio gęstościowo osiągalny z , natomiast nie jest bezpośrednio gęstościowo osiągalny z .

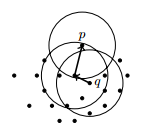


**Rys. 4.** Przykład ilustrujący punkty rdzeniowe i brzegowe

Mówimy, że punkt jest *gęstościowo osiągalny* z punktu względem i , jeżeli istnieje sekwencja punktów takich, że jest *bezpośrednio gęstościowo osiągalny* z . Relacja ta to kanoniczne rozszerzenie relacji bezpośredniej gęstościowej osiągalności, jest tranzytywna lecz nie jest symetryczna. Z tego powodu została wprowadzona symetryczna relacja gęstościowego połączenia.

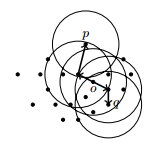
Mówimy, że punkt jest *gęstościowo połączony* z punktem względem i , jeżeli istnieje punkt taki, że i są gęstościowo osiągalne z względem i .

Na Rys. 5 przedstawiono klaster, w którym zaznaczono pewien punkt brzegowy , punkt rdzeniowy , okręgami zaznaczono otoczenia epsilonowe Eps pewnych punktów a wynosi 5. Analiza rysunku pozwala zauważyć, że punkt jest gęstościowo osiągalny z , natomiast jest gęstościowo osiągalny z .



**Rys. 5.** Przykład ilustrujący relację gęstościowej osiągalności

Na Rys. 6 przedstawiono klaster, w którym zaznaczono punkty brzegowe i oraz punkt . Okręgami wyznaczono otoczenia epsilonowe punktów, a wynosi 5. Studium rysunku pozwala spostrzec, że punkty i są gęstościowo osiągalne z , czli punkty i są gęstościowo połączone.



**Rys. 6.** Przykład ilustrujący relację gęstościowej łączności

Wszystkie terminy niezbędne do przedstawienia gęstościowego pojęcia grupy zostały już wprowadzone. Niech D jest zbiorem punktów. Grupą G względem i nazywamy niepusty zbiór D spełniający następujące warunki:

1. : jeśli i jest gęstościowo osiągalne z względem i , wtedy .
2. : jest gęstościowo połączone z względem i .

Niech będą grupami zbioru punków D względem i , . *Szumem* nazywamy podzbiór punktów zbioru D nie należących do żadnej z grup czyli:

Algorytm DBSCAN iteruje wejściowy zbiór punktów i uruchamia procedurę wyznaczania nowej grupy *ExpandCluster* dla każdego punktu, który nie został jeszcze przypisany do którejś z grup lub zidentyfikowany jako szum. *ExpandCluster* w pierwszej kolejności wyznacza otoczenie epsilonowe danego punktu i buduje nową grupę jeśli ów punkt jest punktem rdzeniowym, w przeciwnym przypadku oznacza go jako szum. Proces tworzenia nowej grupy rozpoczyna się od dodania do niej punktów należących do otoczenia epsilonowego danego punktu. Następnie wszystkie punkty epsilonowego sąsiedztwa[[4]](#footnote-5) dodawane są do zbioru ziaren *seeds* zawierającego punkty, które potencjalnie mogą rozszerzyć budowaną grupę. Algorytm iteruje zbiór seeds wyznaczając epsilonowe otoczenie dla każdego jego punktu. Jeżeli dany punkt jest punktem rdzeniowym, to wszystkie punkty należące do jego otoczenia epsilonowego, które nie mają przypisanej żadnej grupy również dodawane są do nowoutworzonej grupy. Te z nich, które nie są oznaczone jako szum dodawane są do zbioru *seeds*, ponieważ mogą rozszerzyć tworzoną grupę. Na wydruku 1 wyżej opisany algorytm został zapisany w formie pseudokodu.

DBSCAN(SetOfPoints, Eps, MinPts)

// SetOfPoints is UNCLASSIFIED

ClusterId := nextId(NOISE);

FOR i FROM 1 TO SetOfPoints.size DO

Point := SetOfPoints.get(i);

IF Point.ClId = UNCLASSIFIED THEN

IF ExpandCluster(SetOfPoints, Point, ClusterId, Eps, MinPts) THEN

ClusterId := nextId(ClusterId);

END IF;

END IF;

END FOR;

END; //DBSCAN

ExpandCluster(SetOfPoints, Point, ClId, Eps, MinPts) : Boolean

seeds := SetOfPoints.regionQuery(Point, Eps);

IF seeds.size < MinPts THEN // no core point

SetOfPoints.changeClId(Point, NOISE);

RETURN false;

ELSE // all points in seeds are density reachable from Point

SetOfPoints.changeClId(seeds, ClId);

seeds.delete(Point);

WHILE seeds <> Empty DO

currentP := seeds.first();

result := SetOfPoints.regionQuery(currentP, Eps);

IF result.size >= MinPts THEN

FOR I FROM 1 TO result.size DO

resultP := result.get(i);

IF resultP.ClId IN (UNCLASSIFIED, NOISE) THEN

IF resultP.ClId = UNCLASSIFIED THEN

seeds.append(resultP);

END IF;

SetOfPoints.changeClId(resultP, ClId);

END IF;

END FOR;

END IF;

seeds.delete(currentP);

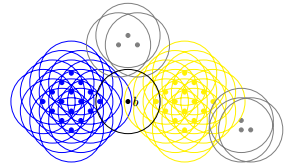
END WHILE;

RETURN true;

END IF;

END; //ExpandCluster

Dokładna analiza kodu pozwala zauważyć, że algorytm DBSCAN jest deterministyczny z dokładnością do punktów brzegowych. Nie bierze on pod uwagę, że punkty brzegowe znajdujące się między leżącymi blisko siebie grupami mogą należeć do więcej niż jednej z grup. Taka sytuacja została przedstawiona na Rys. 7



**Rys. 7.** Ilustracja sytuacji, w której przynależność do jednej z grup (niebieskiej bądź żółtej) punktu brzegowego b zależy od kolejności w jakiej DBSCAN będzie badał punkty

Wynik wykonania algorytmu DBSCAN zależy od kolejności przeglądania punktów, ponieważ punkt brzegowy zakwalifikowany do pewnej grupy, w rezultacie rozbudowy kolejnych grup, może zostać przypisany do innych grup. Problem ten może zostać rozwiązany poprzez przechowywanie w każdym punkcie zamiast jednego identyfikatora *clusteId* zbioru identyfikatorów. Jednakże podobnie do autorów algorytmu, problem ten uznaję za pomijalny.

## 3.2. Porównanie algorytmów VP-Tree i TI-k-Neighborhood

Na przykładowych zbiorach danych zbadałem algorytmy *VP-Tree-Index* oraz *TI-k-Neighborhood-Index* z zastosowaniem odległości kosinusowej jako miary podobieństwa. W i na zamieściłem czasy poszukiwań k-sąsiedztwa przez oba algorytmy.

**Tab. 3.** Porównanie wydajności algorytmów VP-Tree-Index i TI-k-Neighborhood-Index. Tabela zawiera czasy wykonania (w sekundach) poszukiwań k=5 sąsiadów w przykładowych zbiorach danych dla 10% losowo wybranych punktów zbioru danych

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **zbiór** | **l.p.** | **VP-Tree-Index** | | | | **TI-k-Neighborhood-Index** | | | | | |
| **wyk. alg.** | **wysz.** | **bud. ind.** | **norm.** | **wyk. alg.** | **wysz.** | **bud. ind.** | **obl. odl.** | **sort.** | **norm.** |
|
| karypis\_sport | 1000 | 1,145 | 1,059 | 0,083 | 0,002 | 1,034 | 1,027 | 0,000 | 0,005 | 0,000 | 0,002 |
| 2000 | 4,252 | 4,084 | 0,164 | 0,004 | 3,519 | 3,510 | 0,000 | 0,005 | 0,001 | 0,004 |
| 4000 | 18,095 | 17,719 | 0,367 | 0,009 | 14,359 | 14,337 | 0,000 | 0,012 | 0,001 | 0,009 |
| 6000 | 41,553 | 40,977 | 0,563 | 0,013 | 34,451 | 34,418 | 0,000 | 0,018 | 0,001 | 0,013 |
| 8000 | 75,349 | 74,594 | 0,737 | 0,018 | 66,656 | 66,607 | 0,000 | 0,029 | 0,003 | 0,017 |
| karypis\_review | 500 | 0,599 | 0,375 | 0,223 | 0,001 | 0,337 | 0,333 | 0,000 | 0,003 | 0,000 | 0,002 |
| 1000 | 1,930 | 1,509 | 0,418 | 0,003 | 1,293 | 1,286 | 0,000 | 0,004 | 0,000 | 0,003 |
| 2000 | 7,335 | 6,598 | 0,731 | 0,007 | 5,894 | 5,878 | 0,000 | 0,008 | 0,001 | 0,007 |
| 3000 | 15,511 | 14,728 | 0,774 | 0,009 | 12,591 | 12,569 | 0,000 | 0,011 | 0,001 | 0,010 |
| 4000 | 26,991 | 26,361 | 0,616 | 0,013 | 21,877 | 21,847 | 0,000 | 0,016 | 0,001 | 0,013 |
| covtype | 10000 | 2,135 | 1,682 | 0,447 | 0,006 | 2,913 | 2,553 | 0,000 | 0,322 | 0,003 | 0,005 |
| 50000 | 16,985 | 14,367 | 2,585 | 0,033 | 37,549 | 34,851 | 0,001 | 2,934 | 0,016 | 0,025 |
| 100000 | 53,101 | 33,074 | 19,969 | 0,059 | 141,164 | 134,889 | 0,001 | 6,105 | 0,038 | 0,050 |
| 300000 | 345,252 | 248,231 | 96,841 | 0,181 | 1128,400 | 1106,590 | 0,003 | 21,494 | 0,147 | 0,150 |
| 500000 | 989,915 | 812,569 | 177,028 | 0,318 | 3034,090 | 2997,940 | 0,003 | 35,633 | 0,275 | 0,250 |
| cup98 | 10000 | 2,637 | 2,214 | 0,417 | 0,006 | 2,868 | 2,414 | 0,000 | 0,447 | 0,003 | 0,005 |
| 30000 | 25,135 | 22,861 | 2,255 | 0,019 | 9,568 | 8,799 | 0,000 | 0,744 | 0,010 | 0,015 |
| 50000 | 62,006 | 56,783 | 5,191 | 0,032 | 20,456 | 18,402 | 0,001 | 2,009 | 0,019 | 0,025 |
| 70000 | 124,572 | 117,935 | 6,592 | 0,045 | 39,183 | 35,322 | 0,001 | 3,796 | 0,029 | 0,035 |
| 90000 | 193,587 | 186,308 | 7,220 | 0,058 | 63,047 | 58,520 | 0,001 | 4,440 | 0,039 | 0,046 |

Z rezultatów przeprowadzonych badań wynika, że algorytm *TI-k-Neighborhood-Index* wyszukuje k-sąsiedztwo szybciej niż *VP-Tree-Index*. Wyjątek stanowi zbiór covtype, w którego przypadku *VP-Tree-Index* wykonuje się szybciej. Z przebiegów wykresów na .**Błąd! Nie można odnaleźć źródła odwołania.** dla zbiorów różnych od covtype można wnioskować, że algorytm *TI-k-Neighborhood-Index* będzie wykonywał się tym szybciej niż *VP-Tree-Index* im większy będzie zbiór, w którym wyszukiwanych jest k-sąsiadów.

**Rys. 8.** Porównanie wydajności algorytmów VP-Tree-Index i TI-k-Neighborhood-Index. Wykresy zawierają czasy wykonania (w sekundach) poszukiwań k=5 sąsiadów w przykładowych zbiorach danych dla 50% losowo wybranych punktów zbioru danych

Z otrzymanych rezultatów eksperymentów wynika, że dla danych tekstowych algorytm wykorzystujący rzutowanie wykonuje się nieznacznie wolniej niż *TI-k-Neighborhood-Index*. W przypadku gęstych zbiorów wyszukiwanie sąsiedztwa z zastosowaniem rzutowania na losowy wymiar i [dmin] wykonują się kilka razy wolniej niż w pozostałych przypadkach. Warto zwrócić uwagę, że dla zbioru cup98 sprawność *k-neighborhood-Index-Projection* z rzutowaniem na [dmax] jest porównywalna ze sprawnością *TI-k-Neighborhood-Index* dla punktów referencyjnych różnych od [max]. Na niekorzyść rzutowania w porównaniu do punktu referencyjnego przemawia zależność tej metody od liczności dziedzin wymiarów. Z otrzymanych rezultatów eksperymentów wynika, że dla danych tekstowych algorytm wykorzystujący rzutowanie wykonuje się nieznacznie wolniej niż *TI-k-Neighborhood-Index*. W przypadku gęstych zbiorów wyszukiwanie sąsiedztwa z zastosowaniem rzutowania na losowy wymiar i [dmin] wykonują się kilka razy wolniej niż w pozostałych przypadkach. Warto zwrócić uwagę, że dla zbioru cup98 sprawność *k-neighborhood-Index-Projection* z rzutowaniem na [dmax] jest porównywalna ze sprawnością *TI-k-Neighborhood-Index* dla punktów referencyjnych różnych od [max]. Na niekorzyść rzutowania w porównaniu do punktu referencyjnego przemawia zależność tej metody od liczności dziedzin wymiarów.

**Tab. 3.** Porównanie wydajności algorytmów VP-Tree-Index i TI-k-Neighborhood-Index. Tabela zawiera czasy wykonania (w sekundach) poszukiwań k=5 sąsiadów w przykładowych zbiorach danych dla 10% losowo wybranych punktów zbioru danych

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| zbiór | l.p. | VP-Tree-Index | | | | TI-k-Neighborhood-Index | | | | | |
| **wyk. alg.** | **wysz.** | **bud. ind.** | **norm.** | **wyk. alg.** | **wysz.** | **bud. ind.** | **obl. odl.** | **sort.** | **norm.** |
|
| karypis\_sport | 1000 | 1,145 | 1,059 | 0,083 | 0,002 | 1,034 | 1,027 | 0,000 | 0,005 | 0,000 | 0,002 |
| 2000 | 4,252 | 4,084 | 0,164 | 0,004 | 3,519 | 3,510 | 0,000 | 0,005 | 0,001 | 0,004 |
| 4000 | 18,095 | 17,719 | 0,367 | 0,009 | 14,359 | 14,337 | 0,000 | 0,012 | 0,001 | 0,009 |
| 6000 | 41,553 | 40,977 | 0,563 | 0,013 | 34,451 | 34,418 | 0,000 | 0,018 | 0,001 | 0,013 |
| 8000 | 75,349 | 74,594 | 0,737 | 0,018 | 66,656 | 66,607 | 0,000 | 0,029 | 0,003 | 0,017 |
| karypis\_review | 500 | 0,599 | 0,375 | 0,223 | 0,001 | 0,337 | 0,333 | 0,000 | 0,003 | 0,000 | 0,002 |
| 1000 | 1,930 | 1,509 | 0,418 | 0,003 | 1,293 | 1,286 | 0,000 | 0,004 | 0,000 | 0,003 |
| 2000 | 7,335 | 6,598 | 0,731 | 0,007 | 5,894 | 5,878 | 0,000 | 0,008 | 0,001 | 0,007 |
| 3000 | 15,511 | 14,728 | 0,774 | 0,009 | 12,591 | 12,569 | 0,000 | 0,011 | 0,001 | 0,010 |
| 4000 | 26,991 | 26,361 | 0,616 | 0,013 | 21,877 | 21,847 | 0,000 | 0,016 | 0,001 | 0,013 |
| covtype | 10000 | 2,135 | 1,682 | 0,447 | 0,006 | 2,913 | 2,553 | 0,000 | 0,322 | 0,003 | 0,005 |
| 50000 | 16,985 | 14,367 | 2,585 | 0,033 | 37,549 | 34,851 | 0,001 | 2,934 | 0,016 | 0,025 |
| 100000 | 53,101 | 33,074 | 19,969 | 0,059 | 141,164 | 134,889 | 0,001 | 6,105 | 0,038 | 0,050 |
| 300000 | 345,252 | 248,231 | 96,841 | 0,181 | 1128,400 | 1106,590 | 0,003 | 21,494 | 0,147 | 0,150 |
| 500000 | 989,915 | 812,569 | 177,028 | 0,318 | 3034,090 | 2997,940 | 0,003 | 35,633 | 0,275 | 0,250 |
| cup98 | 10000 | 2,637 | 2,214 | 0,417 | 0,006 | 2,868 | 2,414 | 0,000 | 0,447 | 0,003 | 0,005 |
| 30000 | 25,135 | 22,861 | 2,255 | 0,019 | 9,568 | 8,799 | 0,000 | 0,744 | 0,010 | 0,015 |
| 50000 | 62,006 | 56,783 | 5,191 | 0,032 | 20,456 | 18,402 | 0,001 | 2,009 | 0,019 | 0,025 |
| 70000 | 124,572 | 117,935 | 6,592 | 0,045 | 39,183 | 35,322 | 0,001 | 3,796 | 0,029 | 0,035 |
| 90000 | 193,587 | 186,308 | 7,220 | 0,058 | 63,047 | 58,520 | 0,001 | 4,440 | 0,039 | 0,046 |

1. kosinusowe sąsiedztwo to sąsiedztwo oparte na podobieństwie kosinusowym [↑](#footnote-ref-2)
2. α-znormalizowany wektor to znormalizowany wektor przemnożony przez stałą α [↑](#footnote-ref-3)
3. Dendrogram to diagram stosowany do prezentacji związków między elementami lub grupami elementów w kształcie przypominający drzewo. [↑](#footnote-ref-4)
4. Epsilonowe sąsiedztwo jest epsilonowym otoczeniem danego punktu bez niego samego. [↑](#footnote-ref-5)